

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

PUB. NO.: 10-124484 [JP 10124484 A]

PUBLISHED: May 15, 1998 (19980515)

INVENTOR(s): ARAKAWA FUMIO

NAKAGAWA NORIO

YAMADA TETSUYA

TOTSUKA YONETARO

APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP
(Japan)

APPL. NO.: 08-273432 [JP 96273432]

FILED: October 16, 1996 (19961016)

INTL CLASS: [6] G06F-017/10; G06F-007/38; G06F-007/52; G06F-009/38;
G06F-009/38; G06T-001/00

JAPIO CLASS: 45.4 (INFORMATION PROCESSING -- Computer Applications); 45.1
(INFORMATION PROCESSING -- Arithmetic Sequence Units); 45.9
(INFORMATION PROCESSING -- Other)

ABSTRACT

PROBLEM TO BE SOLVED: To perform a vector conversion operation or an inner product operation using the floating points at a high speed and with high accuracy.

SOLUTION: The result of an inner product operation or a vector conversion operation is obtained by giving eight floating points to four multipliers 220a to 220d for their parallel operations and adding together these multiplication results in parallel to each other via a single 4-input adder 226. A circuit 227 normalizes and rounds the output of the adder 226. Thus, an inner product can be calculated via the multiplication and the addition which are carried out once in parallel to each other. Then no rounding process is required for the double input in every product sum operation to shorten the latency of the inner product operation and to improve the arithmetic accuracy. Furthermore, just a single circuit suffices for normalization, etc., and accordingly the increase of the circuit scale is minimized. As a result, an inner product operation, etc., using the floating points can be performed at a high speed and with high accuracy.

特開平10-124484

(43) 公開日 平成10年(1998) 5月15日

(51) Int.Cl.⁶
 G 0 6 F 17/10
 7/38
 7/52 3 1 0
 9/38 3 1 0
 3 7 0

F I
 G 0 6 F 15/31 S
 7/38 A
 7/52 3 1 0 C
 9/38 3 1 0 G
 3 7 0 B

審査請求 未請求 請求項の数13 O L (全 21 頁) 最終頁に続く

(21) 出願番号 特願平8-273432

(22) 出願日 平成8年(1996)10月16日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 荒川 文男

東京都国分寺市東恋ヶ窪一丁目280番地

株式会社日立製作所中央研究所内

(72) 発明者 中川 典夫

東京都小平市上水本町五丁目20番1号 株

式会社日立製作所半導体事業部内

(72) 発明者 山田 哲也

東京都国分寺市東恋ヶ窪一丁目280番地

株式会社日立製作所中央研究所内

(74) 代理人 弁理士 玉村 静世

最終頁に続く

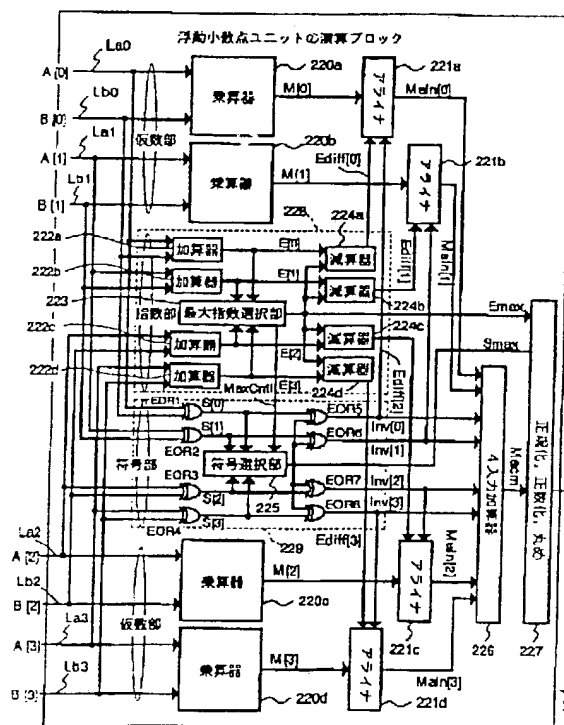
(54) 【発明の名称】 データプロセッサ及びデータ処理システム

(57) 【要約】

【課題】 浮動小数点数を用いたベクトル変換演算や内積演算を高速化及び高精度化する。

【解決手段】 内積演算やベクトル変換演算を行うとき、4個の乗算器(220a~220d)に8個の浮動小数点数を与えて並列動作させ、それによる乗算結果を1個の4入力加算器226で並列的に加算して、演算結果を得る。4入力加算器226の出力に対する正規化や丸めは1個の回路(227)で行う。1回の並列的な乗算及び加算によって内積を求めることができる。2入力に対する積和演算毎に丸めを行う処理も必要ないから、内積演算のレイテンシーが短く、演算精度も高く、正規化等のための回路も1個備えればよいから、回路規模の増大を極力抑えて、浮動小数点による内積演算等を高速に且つ高精度に実行できる。

【図4】



【特許請求の範囲】

【請求項1】 夫々異なるデータ入力信号線群から浮動小数点数の仮数部が供給され、供給された仮数部同士の乗算を行う複数の乗算器と、夫々の乗算器の出力を受けてアライメントシフトを行うアライナと、前記アライナのアライメントシフト数及び正規化前の指数を前記浮動小数点数の指数部に基づいて生成する指数処理部と、前記アライナの出力を並列的に加算する多入力加算器と、前記多入力加算器の出力を前記正規化前の指数に基づいて正規化する正規化器とを含む演算部を浮動小数点ユニットに備えて成るものであることを特徴とするデータプロセッサ。

【請求項2】 前記演算部は更に、夫々の乗算器で乗算される浮動小数点数の符号に応じて、各乗算器の乗算結果に対する符号を生成する符号処理部を含み、前記アライナはアライメントシフト結果を選択的に反転又は非反転で出力するセレクタを有し対応する前記乗算結果に対する符号が負の場合には反転出力を選択し、前記多入力加算器は前記乗算結果に対する符号が負に対応されるアライナの出力に+1を行うキャリーを生成して、負の乗算結果に対し2の補数化処理を行うものであることを特徴とする請求項1記載のデータプロセッサ。

【請求項3】 前記浮動小数点ユニットは更に、前記夫々の乗算器のデータ入力信号線群に接続するリードポートと前記演算部の出力に接続するライトポートとを備えるレジスタファイルを有し、このレジスタファイルは前記リードポートに並列的に接続可能な複数のバンクを有して成るものであることを特徴とする請求項1又は2記載のデータプロセッサ。

【請求項4】 前記浮動小数点ユニットは更に、夫々複数の成分によって表されるデータ同士の内積演算を規定する浮動小数点命令を解読可能な制御部を有し、この制御部は、前記浮動小数点命令を解読して、レジスタファイルが保有するデータの成分を前記信号線群を介して前記演算部に与え、与えられたデータの内積を前記演算部に演算させ、内積の演算結果を前記レジスタファイルに書き込みさせるものであることを特徴とする請求項3記載のデータプロセッサ。

【請求項5】 前記浮動小数点ユニットは更に、夫々複数の成分によって表されるデータと変換行列との行列演算を規定する浮動小数点命令を解読可能な制御部を有し、この制御部は、前記浮動小数点命令を解読して、レジスタファイルが保有するデータの成分と前記変換行列の成分とを讀出して前記信号線群を介し前記演算部に与え、与えられたデータの内積を前記演算部に演算させ、この内積演算の結果を前記レジスタファイルに書き込みさせる一連の演算サイクルを、連続的に複数回繰返し実行させ、連続的に複数回実行される最後の演算サイクルにおける前記レジスタファイルの読出し動作が、最初の演算サイクルにおける内積演算結果を前記レジスタファ

イルに書き込むタイミングよりも早くなるように、前記夫々の演算サイクルのレイテンシーを制御するものであることを特徴とする請求項3記載のデータプロセッサ。

【請求項6】 前記制御部は、前記各演算サイクルにおいて、前記レジスタファイルからの読み出しを双方のバンクに対して並列的に行い、前記レジスタファイルへの書込みを一方のバンクに対して行うためのレジスタ選択制御を行うものであることを特徴とする請求項5記載のデータプロセッサ。

【請求項7】 前記レジスタファイルは各バンクに16個のレジスタを有し、前記乗算器は4個設けられ、前記浮動小数点命令は16ビット固定長命令であることを特徴とする請求項4乃至6の何れか1項記載のデータプロセッサ。

【請求項8】 前記浮動小数点ユニットは更に、角度データに対する正弦及び余弦をテーラ展開による多項近似に従って取得するための浮動小数点命令を解読可能な制御部を有し、この制御部は、レジスタファイルに対するレジスタリードによって角度データを演算ブロックに与え、角度データに対する正弦及び余弦を前記多項近似に従って演算ブロックに演算させ、演算結果をレジスタファイルにライトするものであり、前記角度データは、固定小数点数の小数点以下nビットによって1回転を2のn乗分割して定義するフォーマットを有し、前記多項近似は、前記角度データの小数点以下nビットを中心値とこの中心値に対する差分値に分けて前記角度データに応ずる正弦と余弦の値を演算するものであり、前記多項近似に必要とされる前記中心値に対する正弦又は余弦の値を保有するテーブルを更に備えて成るものであることを特徴とする請求項1乃至3の何れか1項記載のデータプロセッサ。

【請求項9】 前記小数点以下のnビットにおける上位2ビットは角度の象限を指示し、前記制御部は、前記上位2ビットのデコード結果に従って、前記多項近似による演算結果の符号反転とその演算結果を格納するレジスタファイルのレジスタの選択を制御して、前記象限に応ずる正弦及び余弦の値を夫々に割り当てられたレジスタに格納するものであることを特徴とする請求項6記載のデータプロセッサ。

【請求項10】 成分が夫々浮動小数点数で与えられる4×4の内積演算を1個の浮動小数点命令で実行可能な浮動小数点ユニットを含み、この浮動小数点ユニットは演算ブロックと、この演算ブロックに演算対象データを供給し且つ演算ブロックで演算された演算結果データが供給されるレジスタファイルとを含み、前記演算ブロックは、浮動小数点数の仮数部の乗算を行う4個の乗算器と、夫々の乗算器の出力を受けてアライメントシフトを行うアライナと、前記アライナのアライメントシフト数及び正規化前の指数を前記浮動小数点数の指数部に基づいて生成する指数処理部と、前記アライナの出力を並列

的に加算する4入力加算器と、前記4入力加算器の出力を前記正規化前の指数に基づいて正規化する正規化器とを含んで成るものであることを特徴とするデータプロセッサ。

【請求項11】 成分が夫々浮動小数点数で与えられる 4×4 の変換行列と夫々浮動小数点数で与えられる4元の成分を持つベクトルとの積の演算を 4×4 の内積演算を連続4回繰り返して実行する演算を1個の浮動小数点命令で実行可能な浮動小数点ユニットを含み、この浮動小数点ユニットは演算ブロックと、この演算ブロックに演算対象データを供給し且つ演算ブロックで演算された演算結果データが供給されるレジスタファイルとを含み、前記演算ブロックは、浮動小数点数の仮数部の乗算を行う4個の乗算器と、夫々の乗算器の出力を受けてアライメントシフトを行うアライナと、前記アライナのアライメントシフト数及び正規化前の指数を前記浮動小数点数の指数部に基づいて生成する指数処理部と、前記アライナの出力を並列的に加算する4入力加算器と、前記4入力加算器の出力を前記正規化前の指数に基づいて正規化する正規化器とを含み、前記レジスタファイルは夫々16個のレジスタを含む2個のレジスタバンクを有し、前記変換行列は一方のバンクに割り当てられ、前記ベクトルは他方のバンクに割り当てられるものであることを特徴とするデータプロセッサ。

【請求項12】 アドレスバス及びデータバスに結合されたCPUに更に含み、前記浮動小数点ユニットが前記データバスに結合され、前記浮動小数点ユニットは16ビット固定長浮動小数点命令セットを用いて浮動小数点処理を実行し、前記CPUは前記浮動小数点ユニットが浮動小数点処理を実行するための命令とデータを得るのに必要なアドレッシング処理を行うものであり、1個の半導体基板に形成されて成るものであることを特徴とする請求項1乃至11の何れか1項記載のデータプロセッサ。

【請求項13】 請求項12記載のデータプロセッサと、このデータプロセッサに結合され前記CPUによってアクセスされるデータRAMと、前記データRAM及びデータプロセッサに結合された入出力回路とを含んで成るものであることを特徴とするデータ処理システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、内積演算や行列演算に特化したデータプロセッサ、更には3次元グラフィックス制御に最適なデータ処理システムに関し、例えば4元以下の浮動小数点ベクトル又は行列を多用するアプリケーションを実行するデータプロセッサに適用して有効な技術に関するものである。

【0002】

【従来の技術】3次元グラフィックス等では、図形の回転、拡大、縮小、透視投影及び平行移動などに 4×4 の

変換行列を用いた行列演算を多用し、また、受光面の明るさ等を決定するのに内積演算を利用することができる。そのような行列演算や内積演算には積和演算の繰返しが必要になる。また、3次元グラフィックスで扱うデータについては、ハイエンドのシステムでは従来から浮動小数点数が用いられていた。ゲーム機や携帯情報端末等のようなコストの制約が厳しい分野でも、扱うデータは整数から浮動小数点数に移行しつつある。浮動小数点数を用いる方がプログラミングが容易で、高度な処理に向いているからである。

【0003】積和演算器は、単一機能操作として $(A \times B) + C$ の演算を行なうものであり、例えば、Microprocessor Report, Vol. 8, No. 15, November 14, 1994, PP. 6-9PA-8000 Combines Complexity and Speed には積和演算ユニットを備えたプロセッサが示されているが、積和演算ユニットの並列度は2である。

【0004】また、日経エレクトロニクス(日経P B社), 1996. 1. 15(No. 653), pp16-17 には3次元描画機能を1チップに集積した半導体集積回路について記載がある。これには、8個の固定小数点データの演算を1サイクルで実行する積和演算器を組み込んであり、また、 4×4 の行列を使った座標変換を2サイクルで処理できると記載されている。

【0005】

【発明が解決しようとする課題】しかしながら、上記従来技術では、浮動小数点数を用いた 4×4 の行列演算や内積演算などを高速化することについては考慮されていない。本発明者は、浮動小数点数を用いた行列演算や内積演算を高速化することについて検討した。それによれば、浮動小数点数の積和演算器は回路規模が大きいため、単に並列化した場合には、その回路規模の増大が著しく、上記第1の文献にも記載されるように並列度は2程度であって、高速化には限界のあることが明らかにされた。また、第2の文献に記載の内容では 4×4 の行列を使った座標変換を2サイクルで処理でき、ある程度的高速化は実現されているが、ビット数の少ない整数積和演算器を用いる性質上、演算精度は犠牲にならざるを得ないことが明らかにされた。

【0006】本発明の目的は、浮動小数点数を用いた行列演算や内積演算を高速化することができるデータプロセッサを提供することにある。

【0007】本発明の別の目的は、回路規模の増大を極力抑えて行列演算や内積演算を高精度且つ高速に処理できるデータプロセッサを提供することにある。

【0008】本発明の前記並びにその他の目的と新規な特徴は本明細書の記述及び添付図面から明らかになるであろう。

【0009】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記

の通りである。

【0010】すなわち、データプロセッサは、夫々異なるデータ入力信号線群から浮動小数点数の仮数部が供給され、供給された仮数部同士の乗算を行う複数の乗算器と、夫々の乗算器の出力を受けてアライメントシフトを行うアライナと、前記アライナのアライメントシフト数及び正規化前の指数を前記浮動小数点数の指数部に基いて生成する指数処理部と、前記アライナの出力を並列的に加算する多入力加算器と、前記多入力加算器の出力を前記正規化前の指数に基づいて正規化する正規化器とを含む演算部を浮動小数点ユニットに備えて成るものである。

【0011】複数の乗算器による乗算、各乗算器による乗算結果の加算が並列化されることにより、データプロセッサは、浮動小数点による内積演算やベクトル変換演算を高速化できる。また、1回の並列的な乗算及び加算によって内積を求めることができるから、2入力に対する積和演算毎に丸めを行うような処理も必要ないから、内積演算のレイテンシーが短く、演算精度も高く、また、2入力に対する積和演算毎を繰り返す場合のように演算順序が異なると演算結果も相違するという事態も生じない。しかも、データプロセッサは、正規化等のための回路を1個備えればよいから、回路規模の増大を極力抑えて、浮動小数点で内積演算やベクトル変換演算を高速に且つ高精度に行うことが可能になる。

【0012】浮動小数点数の並列的な乗算及び加算における負数に対する処理を能率的に行う様にするには、前記演算部は更に、夫々の乗算器で乗算される浮動小数点数の符号に応じて、各乗算器の乗算結果に対する符号を生成する符号処理部を含み、前記アライナはアライメントシフト結果を選択的に反転又は非反転で出力するセレクタを有し対応する前記乗算結果に対する符号が負の場合には反転出力を選択し、前記多入力加算器は前記乗算結果に対する符号が負に対応されるアライナの出力に+1を行うキャリーを生成して、負の乗算結果に対し2の補数化処理を行うようにするとよい。

【0013】演算部による演算対象データ及び演算結果データはレジスタファイルに一時的に格納する。このとき、乗算器による並列乗算のためには必要なデータ全てがレジスタファイルから複数の乗算器等に並列的に供給されなければならない。このとき、レジスタのポート数およびレジスタ指定フィールドのビット数を増やさずにそのような処理を可能にするために、レジスタファイルをレジスタバンク構成とし、複数のレジスタバンク若しくは単数のバンクのレジスタを、前記乗算器の夫々の入力端子に並列的に接続する様にすればよい。

【0014】内積演算に着目した場合、4元以下の内積を直接求める内積演算命令をデータプロセッサの命令セットに含めるとよい。前記浮動小数点ユニットは、夫々複数の成分によって表されるデータ同士の内積演算を

規定する前記内積演算命令を解読可能な制御部を有し、この制御部は、前記浮動小数点命令を解読して、レジスタファイルが保有するデータの成分を前記信号線群を介して前記演算部に与え、与えられたデータの内積を前記演算部に演算させ、内積の演算結果を前記レジスタファイルに書き込みさせる。

【0015】行列変換演算に着目した場合、変換行列とベクトルとの積を求めるベクトル変換演算命令をデータプロセッサの命令セットに含めるとよい。前記浮動小数点ユニットは、夫々複数の成分によって表されるデータと変換行列との行列演算を規定するベクトル変換演算命令を解読可能な制御部を有し、この制御部は、前記浮動小数点命令を解読して、レジスタファイルが保有するデータの成分と前記変換行列の成分とを読み出して前記信号線群を介し前記演算部に与え、与えられたデータの内積を前記演算部に演算させ、この内積演算の結果を前記レジスタファイルに書き込みさせる一連の演算サイクルを、連続的に複数回繰返し実行させる。このとき、ソースレジスタとディスティネーションレジスタが重なっても正しく動作できる様にするには、連続的に複数回実行される最後の演算サイクルにおける前記レジスタファイルの読み出し動作が、最初の演算サイクルにおける内積演算結果を前記レジスタファイルに書き込むタイミングよりも早くなるように、前記夫々の演算サイクルのレイテンシーを制御すればよい。また、このとき、前記各演算サイクルにおいて、前記レジスタファイルからの読み出しを双方のバンクに対して並列的に行い、前記レジスタファイルへの書き込みを一方のバンクに対して行うようにすれば、浮動小数点レジスタの数の不足を補うことができる。即ち、複数のオペランド（変換行列、ベクトルデータ）を複数バンクに別々に割り当ててレジスタファイルを利用する。

【0016】そのようなレジスタファイルの利用は、16ビット固定長浮動小数点命令のようにレジスタ指定フィールドが限られ、それ故にレジスタの数が制限されるようなアーキテクチャーに対して有用であり、且つ、そのようなリソースの制限されたアーキテクチャーのデータプロセッサにおいて浮動小数点による内積演算やベクトル変換演算を可能にしている。

【0017】また、前記演算部に係数テーブルや乗算器のフィードバック回路を追加することにより、三角関数の近似値を区間分割と高次の多項式展開で求められるようにできる。これによれば、変換行列等に利用される正弦及び余弦を、チップ面積を大幅に増大させることなく得ることができる。

【0018】データプロセッサは、アドレスバス及びデータバスに結合されたCPUを更に含み、前記浮動小数点ユニットが前記データバスに結合され、前記浮動小数点ユニットは16ビット固定長浮動小数点命令セットを用いて浮動小数点処理を実行する。前記CPUは前記浮

動小数点ユニットが浮動小数点処理を実行するための命令とデータを得るのに必要なアドレッシング処理を行う。これにより、浮動小数点ユニットはCPUと同じような高機能なアドレッシングモードをサポートすることを必要とせず、この点においても、浮動小数点命令の16ビット固定長を可能にしている。

【0019】

【発明の実施の形態】

【データプロセッサの構成】図1には本発明の一例に係るデータプロセッサのブロック図が示される。同図に示されるデータプロセッサ1は、32ビットRISC(Reduced Instruction Set Computer)アーキテクチャを有し、16ビット固定長浮動小数点命令を実行する。この実施の態様は特に3次元グラフィックスを十分にサポートする必要があるような機器組み込み制御(例えばビデオ・ゲーム)への応用に有効である。

【0020】このデータプロセッサ1は、浮動小数点ユニット2を有する。浮動小数点ユニット2が演算を行う浮動小数点数は単精度とされる。さらに、データプロセッサ1は中央処理装置(CPU)3を有し、このCPU3は整数を処理する能力を持つ整数ユニットとされる。前記CPU3は32ビットデータバス4を介して前記浮動小数点ユニット2に結合されている。CPU3及び浮動小数点ユニット2は命令バス5を介して命令キャッシュユニット6から命令を取り込む。命令アドレスはCPU3から命令キャッシュユニット6に与えられる。データキャッシュユニット7は、前記データバス4に接続され、データアドレスバス8を介してCPU3からデータアドレスが供給される。前記データキャッシュユニット7及び命令キャッシュユニット6は、夫々図示を省略するキャッシュコントローラ及びキャッシュメモリを備えている。前記命令キャッシュユニット6及びデータキャッシュユニット7はデータ信号やコントロール信号を含むキャッシュバス13を介してバスコントローラ9に接続される。命令キャッシュユニット6におけるキャッシュミス等に起因する外部アクセスのための命令アドレスは前記バスコントローラ9に与えられる。また、データキャッシュユニット7におけるキャッシュミス等に起因する外部アクセスのためのデータアドレスは前記バスコントローラ9に与えられる。バスコントローラ9はそれら命令アドレス又はデータアドレスに従って、代表的に図示されたアドレスピン及びデータピン等に結合される外部メモリなどをアクセスするために外部バスサイクルを起動する。また、バスコントローラ9にはタイマやシリアルコミュニケーションインタフェースコントローラ等の周辺回路10が周辺バス11を介して接続されている。図1に示されるデータプロセッサは、単結晶シリコンのような1個の半導体基板に形成されている。

【0021】前記浮動小数点ユニット(以下単にFPUとも称する)2は浮動小数点処理のためにメモリからデ

ータ又は命令を要求することになる。この実施の態様において、前記FPU2は、データキャッシュユニット7のキャッシュメモリにデータをストアし又は当該キャッシュメモリからデータを獲得するためのメモリアドレッシング能力を持っていない。これは、FPU2のメモリ・アドレッシング回路の必要性を取り除くことによってチップ面積を節約するためである。それに代えて、CPU3はFPU2に代わってキャッシュメモリなどをアドレッシングする機能を有する。したがって、FPU2若しくは浮動小数点命令は、CPU3と同様の強力なアドレッシングモードをサポートする必要はなく、その機能を全てCPU3が負担する。CPU3はFPU2のためにメモリからデータのフェッチを行うだけでなく、CPU3はまた、FPU2のために浮動小数点命令を含む全ての命令をメモリからフェッチする。命令はCPU3とFPU2の双方に取り込まれてデコードされる。CPU3は、デコードした命令がCPU命令である場合にはそれによって指示される整数処理を実行する。また、CPU3は、デコードした命令がFPU命令である場合には、FPU2に代わって実行すべきアドレッシング処理などを行う。FPU2は、デコードした命令がFPU命令である場合にはそれによって指示される浮動小数点処理を実行する。また、FPU2は、デコードした命令がCPU命令である場合にはその命令を無視する。

【0022】図2には前記データプロセッサの主なレジスタ構成が示される。CPUレジスタは16本の汎用レジスタr0~r15と、それに対するカーネルバンクレジスタk0~k7を有する。例えばカーネルバンクレジスタk0~k7は、例外発生時に、汎用レジスタr0~r7の退避に代え、バンク切換え制御によって利用される。

【0023】FPUレジスタはフロントバンクとバックバンクを有するバンクレジスタ構成とされる。フロントバンクは16本のレジスタf0~f15を有する。バックバンクはそれに対応する16本のレジスタb0~b15を有する。何れのバンクのレジスタを利用するかは、例えばコントロールレジスタの制御ビットの値によって決定される。FPUレジスタの場合には更に、特定の浮動小数点命令を実行するとき双方のバンクのレジスタをソースレジスタ及びディスティネーションレジスタとして利用する。その詳細については後述する。フロントバンクのレジスタf0~f15は、単精度フォーマットの浮動小数点数に対しては16本のレジスタとされ、倍精度フォーマットの浮動小数点数に対しては8本のレジスタ(d0, d2, d4, d8, d10, d12, d14)とされ、各成分が単精度フォーマットで与えられる4成分を持つベクトルデータに対しては4本のベクトルレジスタ(v0, v4, v8, v12)とされる。

【0024】また、FPU2とCPU3によって共有されるコミュニケーションレジスタFPUcを有する。こ

のレジスタFPU Cは、CPU 3とFPU 2との間でのデータの受け渡しを高速化するために設けられている。尚、前記各レジスタは32ビット構成である。

【0025】【FPUの構成】図3には前記FPU 2のブロックダイアグラムが示される。FPU 2は、転送ブロック20、レジスタファイル21、演算ブロック22及び制御部23によって構成される。演算ブロック22はその詳細を後述するように行列演算やベクトル演算の高速化を実現した積和演算回路の構成を有する。レジスタファイル21は図2で説明したFPUレジスタを含み、演算ブロック22に対しては8個のリードポートA[0], B[0], A[1], B[1], A[2], B[2], A[3], B[3]を有し、演算ブロック22からデータを受けるライトポートXを有する。転送ブロック20はレジスタファイル21のリードポートCから出力されるデータをデータバス4に供給するバスドライバ200、データバス4からの入力又はレジスタファイル21からの出力を選択してレジスタファイル21のライトポートYに供給するセクタ201を有する。制御部23は、命令バス5から供給された命令をデコードし、そのデコード結果に従って転送ブロック20、レジスタファイル21及び演算ブロック22を制御する。バスドライバ200及びセクタ201の制御信号BusDrv及びLoadCntlも制御部23で形成される。

【0026】図4には演算ブロック22の一例が示される。演算ブロック22は、レジスタファイルのリードポートA[0], B[0], A[1], B[1], A[2], B[2], A[3], B[3]（それらリードポートを単にA[n], B[n]とも記す）に夫々個別に結合する信号線群La0, Lb0, La1, Lb1, La2, Lb2, La3, Lb3（それら信号線群を単にLai, Lbiとも記す）を有する。4個の乗算器220a~220dには夫々の信号線群Lai, Lbiを介して、浮動小数点数の仮数が、乗数及び被乗数として供給される。乗算器220a~220dは、夫々に供給された乗数及び被乗数を乗算して、その積M[0], M[1], M[2], M[3]（以下単にM[n]とも記す）を出力する。積M[0], M[1], M[2], M[3]は、夫々に対応されるアライナ221a~221dに供給される。

【0027】夫々の信号線群Lai, Lbiに供給される浮動小数点数の指数部は夫々加算器222a~222dに供給される。また、夫々の信号線群Lai, Lbiに供給される浮動小数点数の符号部は夫々排他的論理和ゲートEOR1~EOR4に供給される。

【0028】前記加算器222a~222d、最大指数選択部223及び減算器224a~224dは指数処理部228を構成する。加算器222a~222dは乗数と被乗数に対応される浮動小数点数の指数部を加算してその和E[0], E[1], E[2], E[3]を出力する。最大指数選択部223は、前記指数の和E[0], E[1],

E[2], E[3]の内から最大のもをE_{max}として選択する。減算器224a~224dはE_{max}からE[0], E[1], E[2], E[3]を減算して差分Ediff[0], Ediff[1], Ediff[2], Ediff[3]（以下単にEdiff[n]とも記す）を得る。前記差分Ediff[0], Ediff[1], Ediff[2], Ediff[3]は、前記アライナ221a~221dによるアライメントシフト数を制御する。したがって、各アライナ221a~221dの出力Maln[0], Maln[1], Maln[2], Maln[3]（以下単にMaln[n]とも記す）は、最大指数E_{max}に応じた桁位置を持つことになる。このように、指数部228は、前記差分Ediff[0], Ediff[1], Ediff[2], Ediff[3]によって前記アライナ221a~221dのアライメントシフト数を決定すると共に、正規化前の指数E_{max}を浮動小数点数の指数部に基づいて生成する。

【0029】前記排他的論理和ゲートEOR1~EOR4、排他的論理和ゲートEOR5~EOR8及び符合選択部225は符号処理部229を構成する。前記排他的論理和ゲートEOR1~EOR4は乗数と被乗数に対応される浮動小数点数の符号部を入力して、乗数と被乗数の積の符号を判定する。判定された符号S[0], S[1], S[2], S[3]は符合選択部225にて前記E_{max}に応ずる一つがS_{max}として選択される。前記符号S[0], S[1], S[2], S[3]は代表符号S_{max}との一致が排他的論理和ゲートEOR5~EOR8によって判定される。その判定結果Inv[0], Inv[1], Inv[2], Inv[3]（以下単にInv[n]とも記す）は対応するアライナ221a~221dに供給され、例えば判定結果Inv[0], Inv[1], Inv[2], Inv[3]が論理値“1”の場合にはアライナ221a~221dは、対応する積M[n]を反転して出力Maln[n]を形成する。これは積M[n]を2の補数に変換するための前処理とされる。このように、符号処理部229は、夫々の乗算器220a~220dで乗算される浮動小数点数の符号に応じて、正規化前の符号S_{max}及びこの符号S_{max}に対する各乗算器の乗算結果に対する符号Inv[n]を生成する。

【0030】4入力加算器226は、前記アライナ221a~221dの出力Maln[n]を並列的に入力して加算する。4入力の並列加算処理に際して、前記符号Inv[n]が供給される。詳細は後述するが、4入力加算器226は、前記2の補数化の前処理が行われている出力Maln[n]に対してその最下位に+1するための処理を前記符号Inv[n]に基づいて行う。

【0031】4入力加算器226の出力Macmは、正規化、正数化及び丸め処理回路227に供給される。この回路227は、前記正規化前の指数E_{max}と加算出

力 Ma_{cm} と符号 S_{max} とに基づいて正規化および正数化を行い、単精度浮動小数点フォーマットに適合する丸めを行って、浮動小数点数を得る。これによって得られる浮動小数点数は、 $A[0] \cdot B[0] + A[1] \cdot B[1] + A[2] \cdot B[2] + A[3] \cdot B[3]$ の積和演算結果とされる。

【0032】図5には最大指数選択部223の一例が示される。前記 $E[1]$ と $E[0]$ が大小比較器2230によって比較され、大きい方がセクタ2231で選択される。同様に、 $E[3]$ と $E[2]$ が大小比較器2232によって比較され、大きい方がセクタ2233で選択される。双方のセクタで選択されたものは、更に大小比較器2234で比較され、大きい方がセクタ2235で選択される。セクタ2235の出力が前記正規化前の指数 E_{max} とされる。

【0033】図6には符号選択部225の一例が示される。セクタ2250は前記 $S[1]$ 又は $S[0]$ を選択し、セクタ2251は前記 $S[3]$ 又は $S[2]$ を選択し、セクタ2252はセクタ2250の出力又はセクタ2251の出力を選択する。セクタ2250～2252の選択制御信号は前記大小比較器2230、2232、2234の比較判定結果信号 $Ma_{x}C_{nt1}$ とされ、これによって、 E_{max} として選択された指数に係る浮動小数点数の符号部が、前記正規化前の符号 S_{max} として選択される。

【0034】図7にはアライナ221a(221b～221d)の一例が示される。シフタ2210は $M[n]$ を入力し、 $Ediff[n]$ によってアライメントシフト数(シフトビット数)が制御される。シフタ2210の出力はインバータ2211で反転され、インバータ2211の出力又はシフタ2210の出力が $Inv[n]$ によってセクタ2212で選択され、選択された値が $Maln[n]$ とされる。

【0035】図8には4入力加算器226の一例が示される。この4入力加算器226は、桁上げ抜きの和(和出力)と桁上げ(キャリー出力)とをキャリー保存加算器アレイ2260で別々に求め、キャリー伝播加算器2261で最終の和を得る時点まで桁上げの伝播を遅延させる回路形式を有する。この4入力加算器226によって得られる和 Ma_{cm} は入力ビット数に対して最大2ビット増える場合があるから、4入力加算器226に入力される積 $Maln[n]$ は予じめ2ビット符号拡張が施されてキャリー保存加算器アレイ2260に供給される。

【0036】図8において、前記2の補数化のための後処理(+1)は3ビットのキャリー信号 $Cin[0]$ 、 $Cin[1]$ 、 $Cin[2]$ によって行われる。前述の説明から明らかなように、符号選択部225は $S[n]$ の内のどれか一つを選択するから、 $Inv[n]$ のうちの少なくとも一つは必ず論理値“0”にされる。従っ

て、2の補数化の対象は $Maln[n]$ の内の3個以下にしかならない。これを検出するのがORゲート2262、ORゲート2263、AND・ORゲート2264であり、図9には $Inv[n]$ の値に対して $Cin[0]$ 、 $Cin[1]$ 、 $Cin[2]$ の採り得る値が示されており、これによっても明らかなように、 $Cin[2]$ は $Inv[2]$ と $Inv[3]$ の少なくとも一方が論理値“1”のときに論理値“1”にされ、 $Cin[1]$ は $Inv[1]$ と $Inv[0]$ の少なくとも一方が論理値“1”のときに論理値“1”にされ、 $Cin[0]$ は $Inv[1]$ と $Inv[0]$ 或いは $Inv[2]$ と $Inv[3]$ が共に論理値“1”のときに論理値“1”にされる。

【0037】図10には前記キャリー保存加算器アレイ2260とキャリー伝播加算器2261の詳細な論理構成の一例が示される。前記キャリー保存加算器アレイ2260は、特に制限されないが、複数個の4-2コンプレッサ(4-2COMP)2265によって構成される。夫々の4-2コンプレッサ2265は図11の(A)に例示されるように5入力($I1 \sim I4$, Ci)と3出力(S , C , Co)を有する。キャリー出力 Co は隣の上位ビットのキャリー入力 Ci に接続するため、4-2コンプレッサ2265は4個のビット $I1 \sim I4$ を加算する。 S はその加算出力、 C はその加算によって生ずるキャリー出力である。4-2コンプレッサ2265の中では、 Co が Ci に依存しないので、見掛け上、キャリー伝達はない。例えば1個の4-2コンプレッサ2265は、図11の(B)に例示されるように、2個の全加算器によって構成することができる。全加算器は、特に制限されないが、(C)に例示されたマルチプレクサMUXを3個用いて構成される。尚、4-2コンプレッサについて記載された文献としては信学技報(電子情報通信学会) TECHNICAL REPORT OF IEICE ICD94-135, DSP94-91(1994-10)の「バストランジスタ・マルチプレクサを適用した校則54×54ビット乗算器(第73～79頁)がある。

【0038】図10において、夫々の4-2コンプレッサ2265には、前記アライナ出力 $Maln[0] \sim Maln[3]$ における同一桁位置のビットが下位側から順次4ビット単位で供給されている。 $Maln[0]0 \sim Maln[3]0$ は $Maln[0] \sim Maln[3]$ における最下位の4ビットを意味している。前記キャリー信号 $Cin[3]$ は最下位の4-2コンプレッサ2265のキャリー入力端子 Ci に与えられる。前記キャリー伝播加算器2261は複数個の全加算器2261によって構成され、キャリー出力は上位の全加算器のキャリー入力とされる。全加算器の一方の加算入力4-2コンプレッサ2265の和出力 S とされ、もう一方の加算入力は一つ上位に配置された4-2コンプレッサ2265のキャリー出力 C とされる。前記キャリー信号 Cin

〔2〕は最下位の全加算器の一方の加算入力信号として与えられ、前記キャリー信号Cin〔1〕は最下位の全加算器のキャリー入力信号として与えられる。

【0039】図12にはレジスタファイルの一例ブロックダイヤグラムが示され、図13にはレジスタファイルを構成する各レジスタグループの構成が示され、図14にはレジスタグループの各レジスタ回路の構成が示される。

【0040】レジスタファイル21は、特に制限されないが、図12に示されるように4個のレジスタグループFR-Gr.〔0〕～FR-Gr.〔3〕を有し、各レジスタグループFR-Gr.〔m〕は図13に示されるように4個のレジスタ回路FR〔m〕、FR〔m+4〕、FR〔m+8〕、FR〔m+12〕を有する。図13においてmは0～3の整数である。夫々のレジスタ回路は図14に示されるように、フロントバックとバックバンクを構成するための一対のレジスタFRJ〔n〕、FRK〔n〕を有する。図14においてnは0～15の整数である。レジスタFRJ〔n〕、FRK〔n〕に対する書込み動作の指示は信号Write〔n〕によって与えられる。書込み対象とされるレジスタは信号Bankによって何れか一方が選択される。レジスタFRJ〔n〕、FRK〔n〕の出力と端子P〔n〕、Q〔n〕の対応は信号BankによってセクタSL1、SL2で交互に切換え可能にされる。図13に示されるように1個のレジスタグループにおいて、4個のレジスタ回路の端子P〔m〕、P〔m+4〕、P〔m+8〕、P〔m+12〕は、2ビットの信号ReadAによりセクタSL3で何れか1個が選択されることによって端子R〔m〕に接続可能にされ、同様に2ビットの信号ReadBによりセクタSL4で何れか1個が選択されることによって端子B〔m〕に接続可能にされる。レジスタグループの端子Q〔m〕、Q〔m+4〕、Q〔m+8〕、Q〔m+12〕は信号ReadAによって制御されるセクタSL5～SL8でレジスタグループ単位に選択される。前記セクタSL5、SL6、SL7、SL8の出力と夫々のレジスタグループの出力R〔3〕、R〔2〕、R〔1〕、R〔0〕とは信号ReadTypeによって制御されるセクタSL9、SL10、SL11、SL12で選択され、選択されたものがリードポートA〔3〕、A〔2〕、A〔1〕、A〔0〕の出力とされる。したがって、リードポートA〔3〕、A〔2〕、A〔1〕、A〔0〕からは、図15に示されるように、レジスタグループ単位で4個の浮動小数点レジスタから並列的にデータをリードし、或いは、個々のレジスタグループから1個ずつ並列的にデータをリードすることができる。また、夫々のレジスタグループの出力B〔3〕、B〔2〕、B〔1〕、B〔0〕はそのままリードポートB〔3〕、B〔2〕、B〔1〕、B〔0〕の出力とされる。したがって、リードポートB〔3〕、B〔2〕、B〔1〕、B〔0〕からは、図16に示されるように、個々のレジスタグループから1個ずつ並列的にデータをリードすることができる。前記リードポートCには夫々のレジスタグループの出力B〔3〕、B〔2〕、B〔1〕、B〔0〕が信号ReadCによってセクタSL13で選択されたものが接続される。したがって、図17

に示されるように、信号ReadBとReadCとの状態に応じてレジスタを任意に選択してポートCから読み出すことができる。前記ライトポートX、Yからの入力は信号WriteTypeによって制御されるセクタSLで選択される。

【0041】〔内積演算〕前記FPU2を用いた内積演算について説明する。例えば内積は、図18に示されるように、3次元空間において、ある特定の面に光を当てた時の面の明るさを求めるのに利用できる。FPU2は、ベクトルV1(=〔X1, Y1, Z1, W1〕)とV2(=〔X2, Y2, Z2, W2〕)との内積iを1個の浮動小数点内積演算命令(単に内積演算命令とも称する)ftpr Vn, Vmによって求めることができる。

【0042】前記内積演算命令による処理の概略は図19に示される。例えば、レジスタファイル21のベクトルレジスタV0に〔X1, Y1, Z1, W1〕が、V4に〔X2, Y2, Z2, W2〕がロードされているものとする。前記内積演算命令が制御部23で解读されると、レジスタファイル21のリード動作が制御されて、乗算器220aにX1とX2が、乗算器220bにY1とY2が、乗算器220cにZ1とZ2が、乗算器220dにW1とW2が、夫々並列的に供給される。図19では前記指数処理部や符号処理部などの図示を省略しているが、並列的な乗算結果は前記アライナによるシフトや反転等を経て4入力加算器226で加算され、その加算結果に対して正規化等が行われて内積が得られる。得られた内積は、ベクトルレジスタV0の内、W1の値を保有するレジスタにポートXを介して上書きされる。このように、浮動小数点の積和演算が並列的に行われるので、内積演算を高速化できる。

【0043】図20には前記内積演算命令におけるレジスタファイルの利用に関する仕様の一例が示される。即ち、ベクトルV〔m〕とV〔n〕の内積演算結果を浮動小数点レジスタFR〔n+3〕に格納する。ベクトルV〔n〕の成分は浮動小数点レジスタFR〔n〕、FR〔n+1〕、FR〔n+2〕、FR〔n+3〕にロードされる。ここでnは0、4、8、12の何れかであり、またFR〔n〕は前記フロントバンクのレジスタfnに対応されるものと理解されたい。レジスタファイル21の構成上、レジスタFR〔n〕は、制御信号Bank=0の場合にはレジスタFRJ〔n〕に割り当てられ、制御信号Bank=1の場合にはレジスタFRK〔n〕に割り当てられる。この仕様において、例えば図15のBank=0、ReadA=0でポートAからの出力が指定されたレジスタFRJ〔0〕、FRJ〔1〕、FRJ〔2〕、FRJ〔3〕にベクトルデータV〔n〕を置き、図16のBank=0、ReadB=1でポートBからの出力が指定されたレジスタFRJ〔4〕、FRJ〔5〕、FRJ〔6〕、FRJ〔7〕にベクトルデータV〔m〕を置けば、V〔n〕とV〔m〕の内積演算に必要な8個の成分データ

を並列的に演算ブロックに与えて上述の内積演算を行うことができる。演算に際して実際にどのレジスタをリードするかは内積演算命令のレジスタ指定フィールドで指定される。そのレジスタ指定フィールドにおけるレジスタの指定には、ソースレジスタとディスティネーションレジスタの指定に4ビットを用いる。

【0044】〔ベクトル変換演算〕次に、前記FPU2を用いたベクトル変換演算について説明する。周知の4行4列の変換行列は並進、回転、伸張、及び透視等の変換を表す事ができ、この変換行列とベクトルの積によって、その変換行列が表すベクトル変換を得ることができる。ベクトル変換演算は一般に図21で示されるように表すことができる。Aは変換行列、Pは変換対象とされるデータ、P'は変換後のデータである。そのようなベクトル変換は1個の浮動小数点ベクトル変換演算命令(単にベクトル変換演算命令とも称する) `ft r v b a c k, Vn`によって求めることができる。

【0045】前記ベクトル変換演算命令による処理の概略は図22に示される。例えば、変換行列はバックバンクの16本のレジスタに配置される。そしてベクトルデータ $[X_i, Y_i, Z_i, W_i]$ はフロントバンクを構成するレジスタに格納される。

【0046】このベクトル変換演算命令による処理は、実質的に4回の内積演算を順次繰り返す処理に等しい。即ち、 $[X_i, Y_i, Z_i, W_i] \times [a_{11}, a_{12}, a_{13}, a_{14}]$ を演算してその結果を X_i の領域にライト、 $[X_i, Y_i, Z_i, W_i] \times [a_{21}, a_{22}, a_{23}, a_{24}]$ を演算してその結果を Y_i の領域にライト、 $[X_i, Y_i, Z_i, W_i] \times [a_{31}, a_{32}, a_{33}, a_{34}]$ を演算してその結果を Z_i の領域にライト、 $[X_i, Y_i, Z_i, W_i] \times [a_{41}, a_{42}, a_{43}, a_{44}]$ を演算してその結果を W_i の領域にライト、の処理を順次実行する。夫々の処理は実質的に内積演算処理と同じである。

【0047】前記ベクトル変換演算命令が制御部23で解釈されると、上記最初の内積演算処理を行うためのデータがレジスタファイル21から乗算器220a~220d等に夫々並列的に供給される。図22では同じく前記指数処理部や符号処理部などの図示を省略しているが、並列的な乗算結果は前記アライナによるシフトや反転等を経て4入力加算器226で加算され、その加算結果に対して正規化等が行われて内積が得られる。得られた内積は、 X_i を保有するレジスタにライトされる。このような処理をレジスタファイルのリード対象レジスタとライト対象レジスタを順次変更しながら繰り返す。このように、浮動小数点の内積処理を4回連続的に繰り返すことにより、ベクトル変換の結果を高速に得ることができる。

【0048】図23には前記ベクトル変換演算命令におけるレジスタファイルの利用に関する仕様の一例が示され

る。即ち、ベクトル $V[n]$ と変換行列Matrixとの積をレジスタ $V[n]$ に上書きする。ベクトル $V[n]$ の成分は浮動小数点レジスタ $FR[n]$, $FR[n+1]$, $FR[n+2]$, $FR[n+3]$ にロードされる。ここで n は0, 4, 8, 12の何れかであり、また $FR[n]$ は前記フロントバンクのレジスタ f_n に対応されるものと理解されたい。変換行列はバックバンクを構成するレジスタ $FB[0] \sim FB[15]$ (図2の $b_0 \sim b_{15}$ に対応されるレジスタ) に格納される。レジスタファイル21の構成上、レジスタ $FB[n]$ は、制御信号 $Bank=0$ の場合にはレジスタ $FRK[n]$ に割り当てられ、制御信号 $Bank=1$ の場合にはレジスタ $FRJ[n]$ に割り当てられる。

【0049】この仕様において、変換行列Matrixは、図15において $ReadType=1$ の状態でポートAから並列的に出力され、ベクトル $V[n]$ はポートBから並列的に出力される。例えば最初の内積演算では、図15を参照すれば、 $ReadType=1$, $Bank=1$, $ReadA=0$ によって、 $FRJ[0]$, $FRJ[4]$, $FRJ[8]$, $FRJ[12]$ から変換行列Matrixの第1行目がポートAから出力され、これに並行して、 $Bank=1$, $ReadB=0$ によって、 $FRK[0]$, $FRK[1]$, $FRK[2]$, $FRK[3]$ から変換対象ベクトル $[X_i, Y_i, Z_i, W_i]$ がポートBから出力される。順次これに続く3回の内積演算では、それ毎に、 $ReadA$ による選択を1, 2, 3のように変化させればよい。Bポートからのリード対象レジスタは4回の内積演算処理において同一とされる。

【0050】一つのベクトル変換命令による前記複数回の内積演算処理は図24に示されるようにパイプライン処理で行われる。つまり、1つの命令で、4つのパイプライン処理が実行される。内積演算処理の一つのパイプラインは、レジスタリードステージRR、第1演算ステージF1、第2演算ステージF2、第3演算ステージF3、レジスタライトステージRW、及び図示を省略する命令フェッチステージとされる。命令フェッチステージは当然レジスタリードステージRRの前に配置され、また、レジスタリードステージRRは命令のデコード処理も含むことになる。この例では、乗算から正規化までの演算を3個の演算ステージを経て行うことになる。図24の

(1) のパイプラインで実行される処理は、 $(FB[0], FB[4], FB[8], FB[12]) \times V[n]$ の内積演算を行ってその結果をレジスタ $FR[n]$ にライトし、(2) のパイプラインで実行される演算処理は、 $(FB[1], FB[5], FB[9], FB[13]) \times V[n]$ の内積演算を行ってその結果を $FR[n+1]$ にライトし、(3) のパイプラインで実行される演算処理は、 $(FB[2], FB[6], FB[10], FB[14]) \times V[n]$ の内積演算を行ってその結果を $FR[n+2]$ にライトし、(4) のパイプラインで実行される演算処理は、 $(FB[3], FB[7], FB$

[11], $FR[n+3]$) $\times V[n]$ の内積演算を行ってその結果を $FR[n+3]$ にライトするものとされる。ディスティネーションレジスタ $FR[n]$, $FR[n+1]$, $FR[n+2]$, $FR[n+3]$ は $V[n]$ のソースレジスタでもある。

【0051】このとき、一連の4回の内積演算処理において、先頭のパイプライン(1)におけるレジスタライトRWは、最後のパイプライン(4)におけるレジスタリードRRの後にされている。換言すれば、 $V[n]$ の成分と前記変換行列の成分とをレジスタファイル21から読出して内積演算を行い当該内積演算の結果をレジスタファイルにライトする一連の演算サイクルを、連続的に複数回繰返し実行させるとき、連続的に複数回実行される最後の演算サイクルにおける前記レジスタファイルの読出し動作が、最初の演算サイクルにおける内積演算結果を前記レジスタファイルに書き込むタイミングよりも早くなるように、前記夫々の演算サイクルのレイテンシーが制御される。したがって、ソースレジスタとディスティネーションレジスタが同一レジスタであっても、ソースレジスタから全てのデータリードされるまではライトは行われず、データ $V[n]$ が不所望に失われることはない。

【0052】ベクトル変換演算におけるベクトルデータのソースレジスタとディスティネーションレジスタを同一にしても、動作上支障はない。変換前後のベクトルデータを同一レジスタに配置できるので、ベクトル変換処理を多用するプログラムの作成が容易になる。また、3次元グラフィックス等におけるベクトル変換演算は多数のベクトル若しくは点に対して行われることになる。このとき、変換前後のデータが同一レジスタに配置されれば、16本のフロントバックを構成する浮動小数点レジスタに4個のベクトルデータをロードすれば、ベクトル変換命令を4回連続的に実行することができる。即ち、そのような16本の浮動小数点レジスタに対する演算対象データのロード又は演算結果データのメモリへのストア動作の回数が少なくて済む。これに対して、変換前後のデータを別のレジスタの格納する場合には、一つのベクトル変換命令の実行に8本の浮動小数点レジスタを費やす結果、演算対象データのロードや演算結果データのストア動作の頻度が多くなってしまう。この意味において、ベクトル変換演算におけるベクトルデータのソースレジスタとディスティネーションレジスタを同一にできることは、レジスタ本数が限られた中で、ベクトル変換演算を高速化するのに有用である。

【0053】【正弦余弦演算】前記演算ブロック22においては、前記内積演算用のハードウェアに係数テーブルや乗算器のフィードバック回路を追加することにより、三角関数や平方根の近似値を区間分割と高次の多項式展開で求められるようにすることができる。例えば前記変換行列は回転変換のとき正弦及び余弦を含むことになる。必要な角度の全てについて正弦及び余弦のデータ

テーブルを持つ場合には、それによるチップ面積の増大を無視することはできない。

【0054】ここでは、前記演算ブロック22を利用して正弦と余弦の近似値を求めることについて説明する。以下に説明する構成を付加したFPUは1個の命令で正弦と余弦を並行して演算する正弦余弦命令を実行する。この正弦余弦命令の仕様は図25に示される通り、浮動小数点レジスタ $FR[0]$ にロードされた角度データに対する正弦の値を演算して結果をレジスタ $FR[n]$ にライトし、同じく、レジスタ $FR[0]$ にロードされた角度データに対する余弦の値を演算して結果をレジスタ $FR[n+1]$ にライトする。

【0055】図26には前記角度データのフォーマットが示される。前記角度データは、一つの浮動小数点レジスタの上位16ビットと下位16ビットの境を固定小数点位置とする32ビット固定小数点数によって回転数を表すものとされる。小数点位置を境に上位16ビットは回転数(整数)を与え、小数点位置を境に下位16ビットは1回転を2の16乗分割して定義する。特に、下位16ビットの内の上位2ビットは小数点以下16ビットのデータによって特定される角度が属する象限を意味する。このような角度フォーマットにおいて、例えば 360° は1.0であり、16進数のビットパターンは“00010000”とされる。

【0056】正弦余弦命令は、上記角度フォーマットの角度データに対して、その正弦及び余弦を、テーラ展開による多項近似に従って取得する。図28にはその演算手法が示される。

【0057】前記多項近似は、前記角度データの小数点以下16ビットを中心値 x とこの中心値に対する差分値 $d x$ に分けて前記角度データに応ずる正弦と余弦の値を演算するものであり、前記多項近似に必要とされる前記中心値 x に対する正弦又は余弦の値だけはテーブルとして保有する。前記中心値 x は、レジスタ $FR[0]$ の小数点以下7ビットの最下位を0捨1入した値とする。中心値と角度(ラジアン)との関係は図27に例示されている。差分 $d x$ はレジスタ $FR[0]$ の最下位から10ビットを符号拡張した値とする。多項近似ではテーラ展開を用いるため、角度をラジアンで表現するように、各項の係数が与えられている。図28に示される $S1 \sim S12$ は、乗算器220a(FM0), 乗算器220b(FM1), 乗算器220c(FM2), 乗算器220d(FM3), 4入力加算器226を用いた演算処理の内容を式で示している。 $S9$ において多項近似式による正弦の近似値が求められ(図28にはその多項近似式が示されている)、 $S12$ において多項近似式による余弦の近似値が求められる(図28にはその多項近似式が示されている)。

【0058】前記 $S1 \sim S12$ の演算は(1)～(4)で示されるパイプラインで処理される。前述のように、

角度データの低位16ビットの内の上位2ビットは小数点以下16ビットのデータによって特定される角度が属する象限を意味する。したがって、制御部は、前記上位2ビットのデコード結果に従って（その角度データによって特定される角度が属する象限にしたがって）、前記多項近似によるS10、S12の演算結果の符号反転とその演算結果をレジスタFR[n]又はFR[n+1]のどちらに格納するかを選択を制御して、前記象限に応ずる正弦及び余弦の値を夫々に割り当てられたレジスタFR[n]又はFR[n+1]に格納することになる。象限毎の上記反転動作とレジスタ選択動作は図28に示される通りである。

【0059】図29には前記正弦余弦命令を実行するための係数テーブルと乗算器のフィードバック系を付加した前記乗算器近傍のブロックダイアグラムが示される。図29の回路の基本は図4の演算ブロックであり、図4の演算ブロック22に対して、8ビット及び6ビット符号拡張器300、係数テーブル301、セクタ302～312が追加された点が異なるだけである。レジスタFR[0]の角度データはポートB[0]から与えられる。8ビット及び6ビット符号拡張器300はレジスタFR[0]の最下位10ビットから差分dxを生成する回路である。係数テーブル301は図27に示す中心値に応ずる正弦又は余弦の何れか一方のデータを保有し、角度データの低位16ビットの内の下位5ビットによって指定される角度の正弦及び余弦のデータを出力する。角度データの低位16ビットの上位2ビットは制御部に供給される。制御部は、その2ビットの値に従って、前記4入力加算器による加算出力の選択的な反転と、加算結果を格納するレジスタFR[n]又はFR[n+1]の選択を制御することになる。尚、係数テーブル301が正弦データを持つ場合に、余弦は中心値xの角度を $\pi/2$ から減算した角度でテーブルを参照すればよい。係数テーブルに正弦及び余弦の双方のデータを持ってもよい。

【0060】図30は図28の（1）で示される第1ステップにおけるデータの流れを太い実線によって示す。図31は図28の（2）で示される第2ステップにおけるデータの流れを太い実線によって示す。この図において、乗算器（FMO）220aの乗算結果がセクタ308、311にフィードバックされているが、フィードバックは小数点以下のみとされ（上位は0とする）、+1の効果も得るようにしている。図32は図28の（3）で示される第2ステップにおけるデータの流れを太い実線によって示す。図33は図28の（4）で示される第2ステップにおけるデータの流れを太い実線によって示す。図32及び図33において演算結果を2度利用する場合には乗算器の入力ラッチの更新を抑止してその値を保持する。図30乃至図32に示される演算制御は、正弦余弦命令をデコードする制御部が行う。

【0061】【データプロセッサ1の優位性】上記FP

U2の演算ブロック22は前述のように、内積演算命令やベクトル変換演算命令等の1個の命令を実行するとき、4個の乗算器220a～220dに8個の浮動小数点数を与えて並列動作させ、それによる乗算結果を1個の4入力加算器226で加算して、演算結果を得る。4入力加算器226の出力に対する正規化、正数化及び丸めは1個の回路227によって行う。図34にはその演算処理におけるデータの流れを理解し易いように演算ブロック22の概略を示してある。

【0062】図35には、上記演算ブロック22に対する比較例が示されている。これは、一対の浮動小数点数に対する積和演算器と、その結果に対する正規化、正数化及び丸めのための回路とを2組設けて構成される。図36には上記演算ブロック22に対する別の比較例として、上記積和演算器と正規化、正数化及び丸めのための回路とを4組み設けたものが示されている。何れの比較例も、積和演算器と正規化、正数化及び丸めのための回路とを複数組み並列化したに過ぎない。したがって、4×4の一つの内積を演算する場合には、積和演算、正規化、正数化及び丸めのための複数個の回路を単に並列動作させるだけでは済まない。夫々の演算結果に対する相関を考慮した制御が別に必要とされる。通常は、一組の積和演算器と正規化、正数化及び丸めのための回路とを4回繰返し動作させて内積を求めることになるであろう。ベクトル変換演算の場合には更に多くの演算サイクルが必要になる。積和演算、正規化、正数化及び丸めのための複数組の回路は、パイプラインのような命令実行手法によって、対象の異なる内積演算やその他の浮動小数点命令のために並列動作されるであろう。この意味において、図35及び図36に示される回路構成は、種々の浮動小数点命令の演算処理能力を平均的に向上させ得るという点に特徴がある。

【0063】上記演算ブロック22を用いる場合には、内積演算やベクトル変換演算のための実質的な演算サイクル数を少なくすることができる。すなわち、内積演算やベクトル変換演算の高速化を実現出来る。このように、演算ブロック22の構成は、内積演算やベクトル変換演算の高速化に特化している。また、1回の並列的な乗算及び加算によって内積を求めることができるから、2入力に対する積和演算毎に丸めを行うような処理も必要ない。これにより、内積演算のレイテンシーが短く、演算精度も高く、また、2入力に対する積和演算毎を繰り返す場合のように演算順序が異なると演算結果も相違するという事態も生じない。

【0064】また、正規化、正数化及び丸め回路は積和演算回路と同等の回路規模を有することになるので、図35及び図36のように積和演算、正規化、正数化及び丸めのための回路を複数組単に並列配置した構成では、並列化によって達成しようとする平均的な演算能力の向上に比べて、並列化によるチップ面積の増大が極めて大

きくなる。この意味において、積和演算、正規化、正数化及び丸めのための回路の並列数は2が妥当と考えられている。整数演算だけなら、図37のように積和演算器を4個並列させることも現実的であるが、整数演算の場合にはデータの桁数が限られるために浮動小数点演算に比べて演算精度は低くなってしまふ。図34の演算ブロック22は、正規化、正数化及び丸めのための回路227を1個備えればよい。したがって、データプロセッサは、回路規模の増大を極力抑えて、浮動小数点で内積演算やベクトル変換演算の高速化を実現出来る。

【0065】データプロセッサ1の浮動小数点命令は16ビット固定長であり、それ故に、浮動小数点命令におけるアドレス指定フィールドは限られ、浮動小数点レジスタは16本とされる。このような制約の下において、浮動小数点レジスタをバックバンクとフロントバンクを持つレジスタバンク構成のレジスタファイル21によって構成している。このとき、上記変換行列全体を格納するのに16個のレジスタを消費するため、ベクトル変換演算命令においてはフロントバンクとバックバンクの双方を利用する命令仕様とされている。前述したように、変換行列をバックバンクに配置し、ベクトルデータをフロントバンクに配置する。これにより、浮動小数点命令のビット数とレジスタ本数というリソースの制約下においても、ベクトル変換演算命令の高速実行を保証している。

【0066】また、ベクトル変換演算命令の実行において、前述のように、一連の4回の内積演算処理は、先頭のパイプライン(1)におけるレジスタライトRWが、最後のパイプライン(4)におけるレジスタリードRRの後にされているように、パイプライン化されている。したがって、ソースレジスタとディスティネーションレジスタが同一レジスタであっても、演算対象とされるベクトルデータは不所望に失われない。これにより、多数のベクトル若しくは点に対してベクトル変換が次々に行われるとき、変換前後のデータが同一レジスタに配置されれば、16本のフロントバックを構成する浮動小数点レジスタに4個のベクトルデータをロードすれば、ベクトル変換命令を4回連続的に実行することができ、そのような16本の浮動小数点レジスタに対する演算対象データのロード又は演算結果データのメモリへのストア動作の回数が少なくて済む。この意味において、ベクトル変換演算におけるベクトルデータのソースレジスタとディスティネーションレジスタを同一にできることは、レジスタ本数が限られた中で、ベクトル変換演算を高速化するのに有用である。

【0067】また、前記変換行列は回転変換のとき正弦及び余弦を含むことになる。必要な角度の全てについて正弦及び余弦のデータテーブル301を持つ場合には、それによるチップ面積の増大を無視することはできない。このとき、前記演算ブロック22は4個の乗算器を含んでいるので、それに係数テーブル301や乗算器の

フィードバック回路を追加することにより、三角関数や平方根の近似値を区間分割と高次の多項式展開で求められるようにすることができる。これによって、正弦及び余弦をチップ面積を増大させることなく得ることができる。特に正弦と余弦の多項近似の展開式には類似性があるので、これを利用して正弦及び余弦の値を同時(並列的)に演算するので、個別に求める場合に比べて正弦及び余弦の値を高速に得ることができる。

【0068】図38にはそのようなデータプロセッサを適用したデータ処理システムのブロックダイアグラムが示される。

【0069】同図において1は上記データプロセッサ、401はダイナミック・ランダム・アクセス・メモリ(DRAM)、402はDRAM401に対するアドレスマルチプレクス制御やリフレッシュ制御を行うDRAM制御部、403はSRAMである。SRAM403はデータプロセッサ1の作業領域やデータの一時記憶領域などに利用される。404はデータプロセッサ1のOS(Operating System)などを保有するROMである。405は周辺装置制御部であり、代表的に示された外部記憶装置406及びキーボード407が接続されている。408はフレームバッファ409や図示しない描画及び表示制御論理回路を備えた表示コントローラであり、ディスプレイ410に対する描画制御と表示制御を行う。411は電源回路、412は代表的に示されたバスである。データプロセッサ1は3次元グラフィック処理に多用される内積演算やベクトル変換演算等を浮動小数点で高速に実行することができる。しかも、浮動小数点命令のビット数及びレジスタ本数等の限られたリソースの下で上記効果を得ることができるから、データプロセッサ1のコストも低く抑えられている。したがって、図38のデータ処理システムは、システムのコストを抑えて、3次元グラフィック処理を高精度に且つ高速に行うことができる。したがって、コストの制約は厳しいけれども、高機能及び高速化の要請も無視出来ないような、ゲーム機や携帯情報端末などに適用して優れたデータ処理システムを実現出来る。

【0070】以上本発明者によってなされた発明を実施形態に基づいて具体的に説明したが、本発明はそれ限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。

【0071】例えば、図1では説明していないが、データプロセッサはメモリマネージメントユニットなどのその他の機能ブロックを含むことができる。また、データプロセッサは、スーパーカラーアーキテクチャを採用することができる。例えば、2本のパイプを有する場合、一方のパイプではベクトル変換演算命令などを実行し、他方のパイプではベクトル変換演算命令のためのベクトルデータをメモリからレジスタファイルのロードしたり、ベクトル変換演算の結果をレジスタファイルから

メモリにストアすることができる。

【0072】また、乗算器の並列配置個数は4個以上であってもよい。また、指数処理部や符号処理部の構成、4入力加算器の構成は上記実施例に限定されず適宜変更可能である。

【0073】また、本発明のデータプロセッサはゲーム機や携帯情報端末の制御に適用される場合に限定されず、種々の機器組み込み制御などの用途に広く利用することができる。

【0074】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【0075】すなわち、データプロセッサは、浮動小数点による内積演算やベクトル変換演算の高速化を実現できる。

【0076】データプロセッサは、正規化等のための回路を1個備えればよいから、回路規模の増大を極力抑えて、浮動小数点で内積演算やベクトル変換演算の高速化を実現できる。

【0077】ベクトル変換演算命令においてはフロントバンクとバックバンクの双方を利用する命令仕様とされており、変換行列をバックバンクに配置し、ベクトルデータをフロントバンクに配置することにより、浮動小数点命令のビット数とレジスタ本数というリソースの制約下においても、ベクトル変換演算命令の高速実行を保証できる。

【0078】また、ベクトル変換演算命令の実行において、一連の4回の内積演算処理は、先頭の内積演算処理におけるレジスタライトが、最後の内積処理におけるレジスタリードの後にされるように、各内積処理のレイテンシーを制御するから、ソースレジスタとディスティネーションレジスタが同一レジスタであっても、演算対象とされるベクトルデータは不所望に失われない。これにより、浮動小数点レジスタに対する演算対象データのロード又は演算結果データのメモリへのストア動作の回数が少なく済み、レジスタ本数が限られた中で、ベクトル変換演算を高速化するのに有用である。

【0079】また、前記演算部に係数テーブルや乗算器のフィードバック回路を追加することにより、三角関数の近似値を区間分割と高次の多項式展開で求められるようにすることにより、変換行列等に利用される正弦及び余弦を、チップ面積を大幅に増大させることなく得ることができる。

【0080】データプロセッサは、浮動小数点命令のビット数及びレジスタ本数等の限られたリソースの下において、3次元グラフィック処理に多用される内積演算やベクトル変換演算等を浮動小数点で高速に実行することができるから、前記データプロセッサを適用したデータ処理システムは、システムのコストを抑えて、3次元グ

ラフィック処理を高精度に且つ高速に行うことができる。したがって、コストの制約は厳しいけれども、高性能及び高速化の要請も無視出来ないような、ゲーム機や携帯情報端末などに適用して優れたデータ処理システムを実現出来る。

【図面の簡単な説明】

【図1】本発明の一例に係るデータプロセッサのブロック図である。

【図2】図1のデータプロセッサの主なレジスタ構成の説明図である。

【図3】前記FPUの一例ブロック図である。

【図4】演算ブロックの一例ブロック図である。

【図5】最大指数選択部の一例ブロック図である。

【図6】符合選択部の一例ブロック図である。

【図7】アライナの一例ブロック図である。

【図8】4入力加算器の一例ブロック図である。

【図9】4入力加算器においてInv[n]の値に対してCin[0], Cin[1], Cin[2]の採り得る値を示す説明図である。

【図10】キャリー保存加算器アレイとキャリー伝達加算器アレイの詳細な一例ブロック図である。

【図11】4-2コンプレッサの一例説明図である。

【図12】レジスタファイルの一例ブロック図である。

【図13】レジスタファイルを構成する各レジスタグループの構成説明図である。

【図14】レジスタグループの各レジスタ回路の一例構成図である。

【図15】レジスタファイルのリードポートAの動作態様説明図である。

【図16】レジスタファイルのリードポートBの動作態様説明図である。

【図17】レジスタファイルのリードポートCの動作態様説明図である。

【図18】内積の応用例を示す説明図である。

【図19】内積演算命令による処理の概略を示すブロック図である。

【図20】内積演算命令におけるレジスタファイルの利用に関する仕様の一例説明図である。

【図21】ベクトル変換演算の一般的に示す説明図である。

【図22】ベクトル変換演算命令による処理の概略を示すブロック図である。

【図23】ベクトル変換演算命令におけるレジスタファイルの利用に関する仕様の一例説明図である。

【図24】一つのベクトル変換命令による複数回の内積演算処理のパイプラインを示す説明図である。

【図25】正弦余弦命令の仕様説明図である。

【図26】正弦余弦命令に利用される角度データのフォーマット説明図である。

【図27】多項近似のための中心値と角度（ラジアン）

との関係を示す説明図である。

【図28】多項近似に従った正弦余弦命令による演算処理の流れ図である。

【図29】正弦余弦命令を実行するための係数テーブルと乗算器のフィードバック系を付加した前記乗算器近傍のブロック図である。

【図30】図28の(1)で示される第1ステップにおけるデータの流れを太い実線によって示す説明図である。

【図31】図28の(2)で示される第2ステップにおけるデータの流れを太い実線によって示す説明図である。

【図32】図28の(3)で示される第2ステップにおけるデータの流れを太い実線によって示す説明図である。

【図33】図28の(4)で示される第2ステップにおけるデータの流れを太い実線によって示す説明図である。

【図34】内積演算命令やベクトル変換演算命令を実行するときの演算処理におけるデータの流れを理解し易いように演算ブロック22を概略的に示したブロック図である。

【図35】積和演算器と正規化、正数化及び丸めのための回路とを2組み単に並列配置したものを示す比較説明図である。

【図36】積和演算器と正規化、正数化及び丸めのための回路とを4組み単に並列配置したものを示す比較説明図である。

【図37】整数演算のために積和演算器を4個並列させたものを示す比較説明図である。

【図38】データプロセッサを適用したデータ処理システムの一例ブロック図である。

【符号の説明】

1 データプロセッサ

2 浮動小数点ユニット

3 CPU

4 データバス

5 命令バス

20 転送ブロック

21 レジスタファイル

22 演算ブロック

23 制御部

A[0], A[1], A[2], A[3] リードポート

B[0], B[1], B[2], B[3] リードポート

C リードポート

X, Y ライトポート

f0~f15 フロントバンクを構成するレジスタ

b0~b15 バックバンクを構成するレジスタ

220a~220d 乗算器

221a~221d アライナ

222a~222d 加算器

223 最大指数選択部

224a~224d 減算機

228 指数処理部

EOR1~EOR8 排他的論理ゲート

225 符合選択部

229 符号処理部

226 4入力加算器

227 正規化、正数化及び丸め回路

La0~La3, Lb0~Lb3 信号線群

E_{max} 正規化前の指数

S_{max} 正規化前の符号

Inv[0], Inv[1], Inv[2], Inv[3] 乗算結果に対する符合

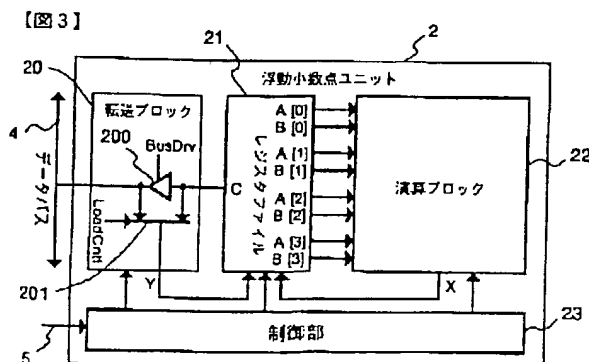
300 8ビット及び6ビット符号拡張器

301 正弦余弦テーブル

403 SRAM

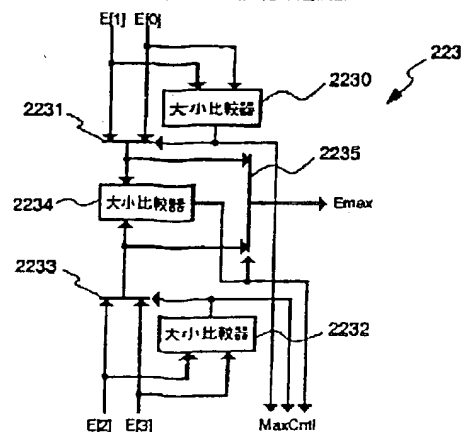
412 バス

【図3】

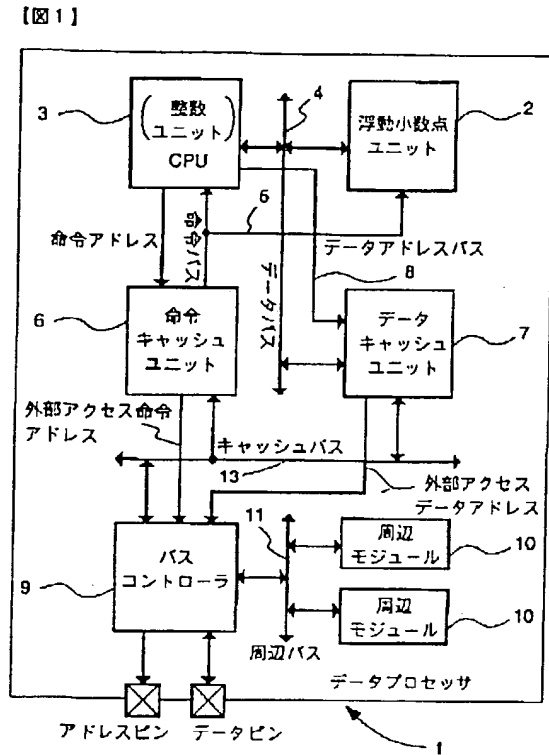


【図5】

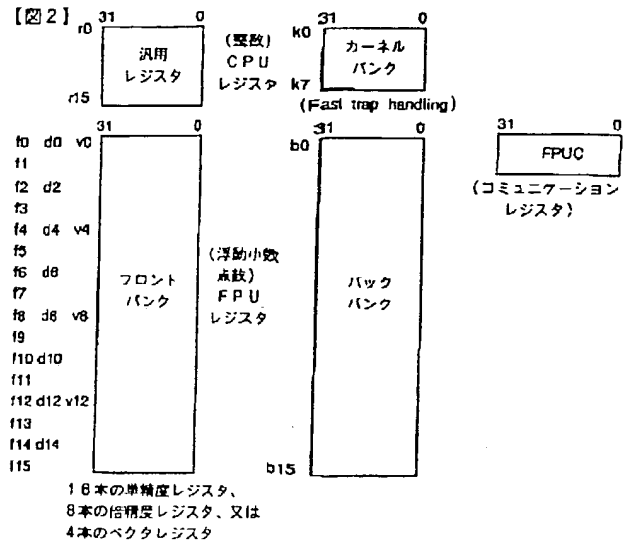
【図5】 演算ブロックの最大指数選択部



【図 1】



【図 2】



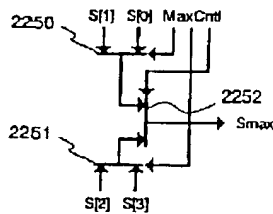
【図 9】

【図 9】 キャリー入力変換論理

Inv	(0)	(1)	(2)	(3)	Cin	(0)	(1)	(2)
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	1	1	0
1	0	0	0	0	1	0	1	0
0	0	1	1	1	1	0	1	1
0	1	0	1	0	0	1	1	1
0	1	1	0	0	0	1	1	1
1	0	1	0	0	0	1	1	1
1	1	0	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1

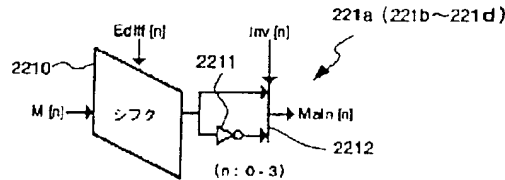
【図 6】

【図 6】 演算ブロックの符号選択部



【図 7】

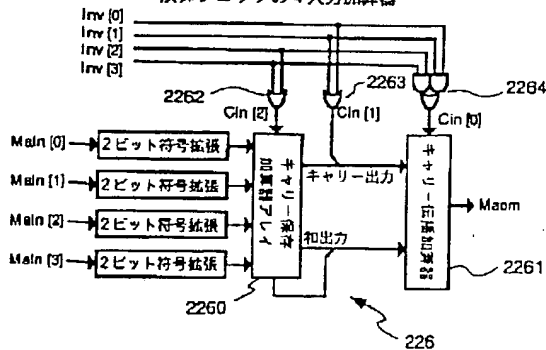
【図 7】 演算ブロックのアライナ



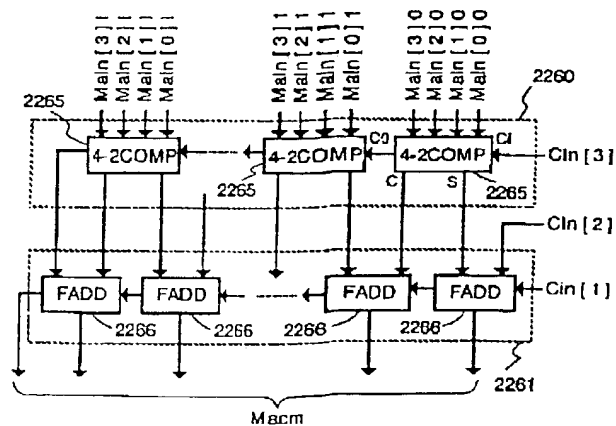
【図 10】

【図 8】

【図 8】 演算ブロックの 4 入力加算器

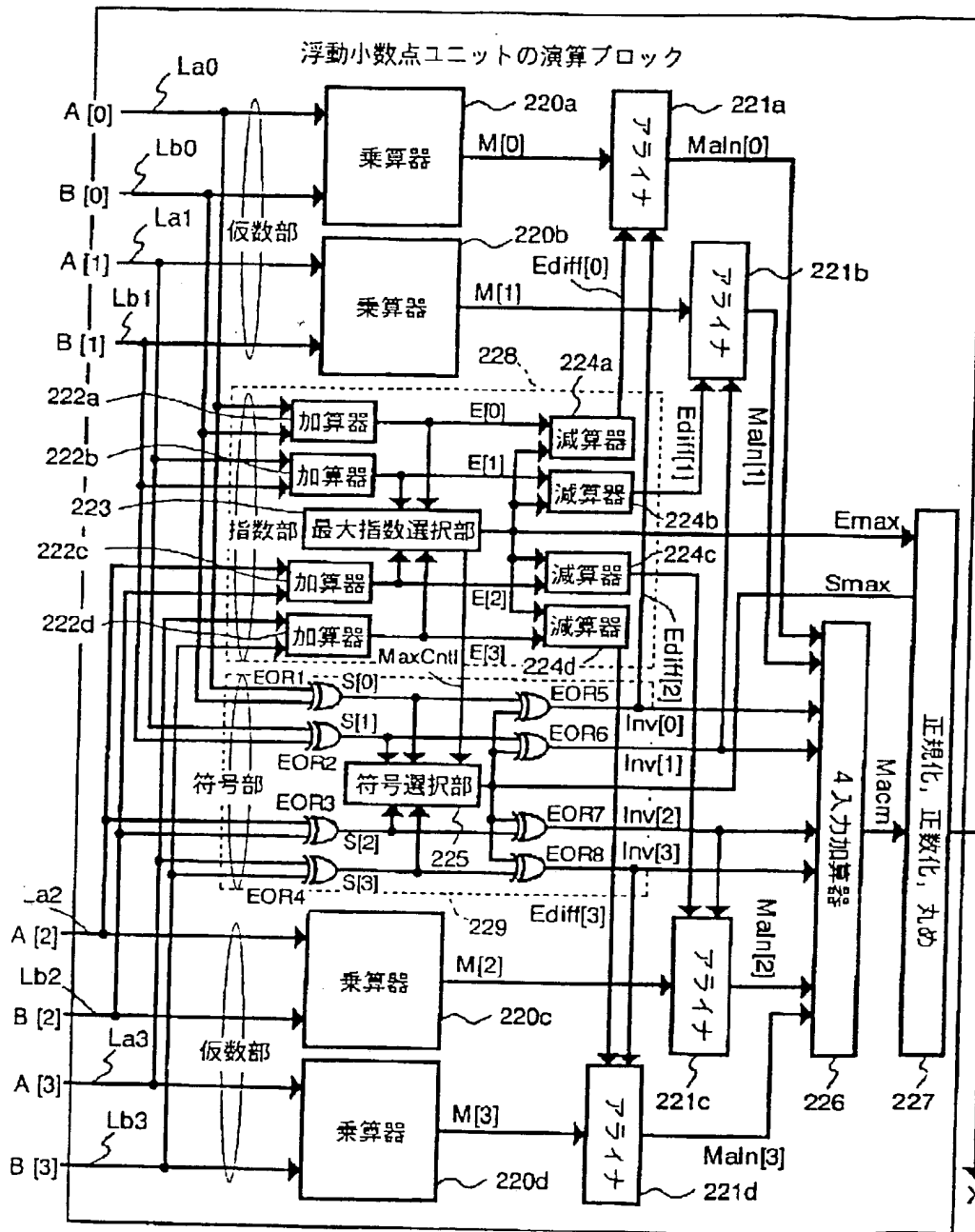


【図 10】



【図 4】

【図 4】

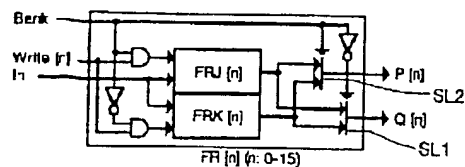


【図 1 4】

【図 2 0】

【図 2 5】

【図 1 4】 レジスタグループの各レジスタ回路の構成図



【図 2 0】 内積命令仕様

$$FR[n+3] = V[m], V[n] \text{ 但し, } V[n] = \begin{pmatrix} FR[n] \\ FR[n+1] \\ FR[n+2] \\ FR[n+3] \end{pmatrix} (n: 0, 4, 8, 12)$$

$$FR[n] = \begin{cases} FRJ[n] & \text{Bank} = 0 \text{ の場合} \\ FRK[n] & \text{Bank} = 1 \text{ の場合} \end{cases}$$

【図 2 5】

正弦余弦命令仕様

$$FR[n] = \sin FR[n]$$

$$FR[n+1] = \cos FR[n]$$

【図 17】

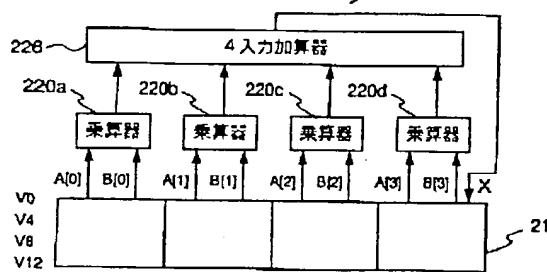
【図 17】

レジスタファイルのリード動作C

Bank	ReadB	ReadC	0	1	2	3
0	0		FRJ [0]	FRJ [1]	FRJ [2]	FRJ [3]
	1		FRJ [4]	FRJ [5]	FRJ [6]	FRJ [7]
	2		FRJ [8]	FRJ [9]	FRJ [10]	FRJ [11]
	3		FRJ [12]	FRJ [13]	FRJ [14]	FRJ [15]
1	0		FRK [0]	FRK [1]	FRK [2]	FRK [3]
	1		FRK [4]	FRK [5]	FRK [6]	FRK [7]
	2		FRK [8]	FRK [9]	FRK [10]	FRK [11]
	3		FRK [12]	FRK [13]	FRK [14]	FRK [15]

【図 19】

【図 19】



【図 23】

【図 23】

ベクトル変換命令仕様

$$V[n] = \text{Matrix} \times V[n]$$

$$\text{但し, Matrix} = \begin{pmatrix} \text{FB}[0] & \text{FB}[4] & \text{FB}[8] & \text{FB}[12] \\ \text{FB}[1] & \text{FB}[5] & \text{FB}[9] & \text{FB}[13] \\ \text{FB}[2] & \text{FB}[6] & \text{FB}[10] & \text{FB}[14] \\ \text{FB}[3] & \text{FB}[7] & \text{FB}[11] & \text{FB}[15] \end{pmatrix}$$

$$\text{FB}[n] = \begin{cases} \text{FRK}[n] & \text{Bank} = 0 \text{ の場合} \\ \text{FRJ}[n] & \text{Bank} = 1 \text{ の場合} \end{cases}$$

【図 26】

【図 26】

正弦余弦命令の角度フォーマット

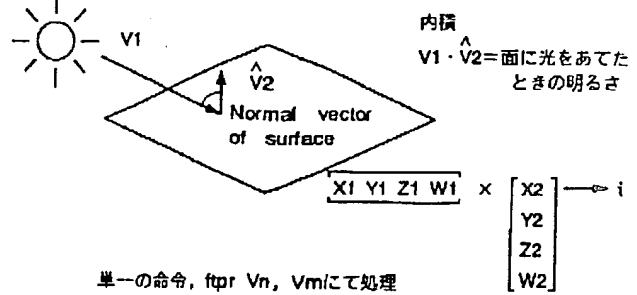
上位16ビット 下位16ビット

小数点

小数点以下16ビットの32ビット固定少数点数で回転数を表す。

【図 18】

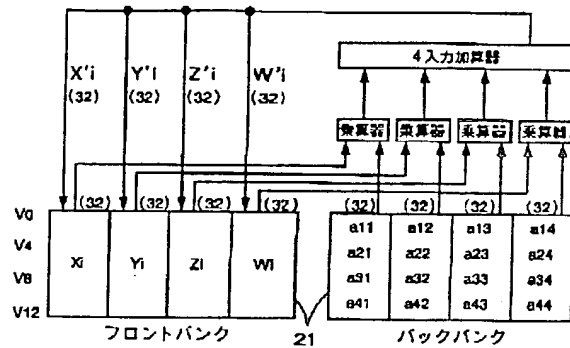
【図 18】



単一の命令, fopr Vn, Vmにて処理

【図 22】

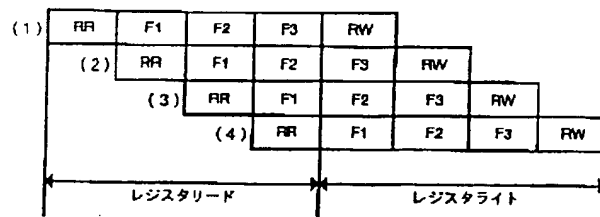
【図 22】



【図 24】

【図 24】

ベクトル変換命令処理パイプライン



- (1) FR[n] = (FB[0] FB[4] FB[8] FB[12]) V[n]
- (2) FR[n+1] = (FB[1] FB[5] FB[9] FB[13]) V[n]
- (3) FR[n+2] = (FB[2] FB[6] FB[10] FB[14]) V[n]
- (4) FR[n+3] = (FB[3] FB[7] FB[11] FB[15]) V[n]

RR: レジスタリードステージ
F1: 第一演算ステージ
F2: 第二演算ステージ
F3: 第三演算ステージ
RW: レジスタライトステージ

【図 27】

【図 27】

角度入力の下位14ビットの内の上位5ビットの値に対応する級数展開の中心値

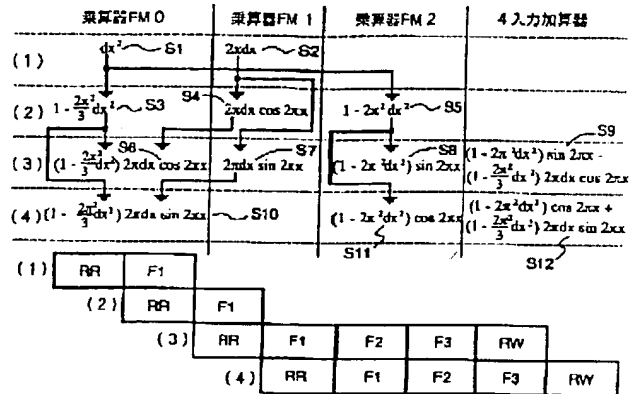
角度入力	級数展開の中心値 内部表現	ラジアン	角度入力	級数展開の中心値 内部表現	ラジアン
1 1 1 1 1	0.000000	0	0 1 1 1 1	0.001000	$\pi/4$
0 0 0 0 0	0.000001	$\pi/32$	1 0 0 0 0	0.001001	$9\pi/32$
0 0 0 0 1	0.000010	$\pi/16$	1 0 0 0 1	0.001010	$5\pi/16$
0 0 0 1 1	0.000011	$3\pi/32$	1 0 0 1 1	0.001011	$11\pi/32$
0 0 1 0 0	0.000100	$\pi/8$	1 0 1 0 0	0.001100	$3\pi/8$
0 0 1 0 1	0.000101	$5\pi/32$	1 0 1 0 1	0.001101	$13\pi/32$
0 0 1 1 0	0.000110	$3\pi/16$	1 0 1 1 0	0.001110	$7\pi/16$
0 0 1 1 1	0.000111	$7\pi/32$	1 0 1 1 1	0.001111	$15\pi/32$

【図 28】

【図 28】

正弦余弦命令の処理フロー

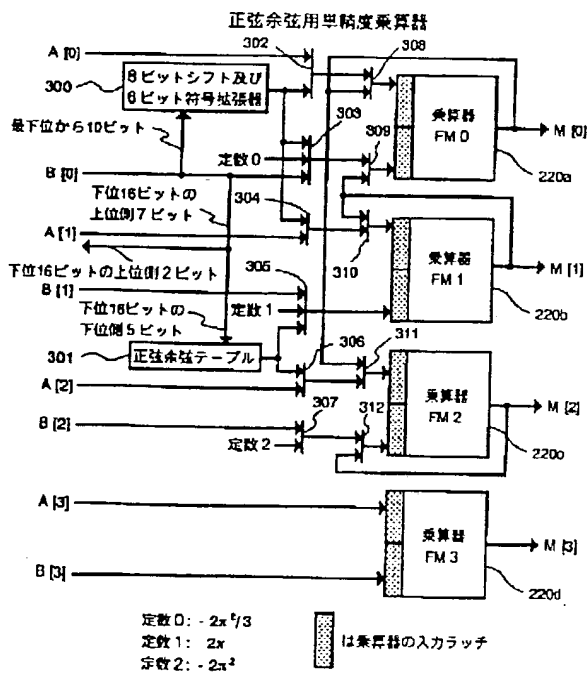
x は FR 関の小数点以下7ビットの最下位を0者1入した値
dx は FR 関の最下位10ビットを符号拡張した値
乗算器FM 0, FM 2 では小数点以下のみフィードバックして上位を0とし、+1と
同じ効果を得る演算結果を2度使用する場合は乗算器の入力ラッチの更新を抑制し
て値を保持する



x + dx が第 I 象限:
ステップ (3) で 4 入力加算器出力を後処理 (正規化, 正数化, 丸め) して FR [n] へ
ステップ (4) で 4 入力加算器出力を後処理して FR [n+1] へ
x + dx が第 II 象限:
ステップ (3) で 4 入力加算器出力を符号反転および後処理して FR [n+1] へ
ステップ (4) で 4 入力加算器出力を後処理して FR [n] へ
x + dx が第 III 象限:
ステップ (3) で 4 入力加算器出力を符号反転および後処理して FR [n] へ
ステップ (4) で 4 入力加算器出力を符号反転および後処理して FR [n+1] へ
x + dx が第 IV 象限:
ステップ (3) で 4 入力加算器出力を後処理して FR [n+1] へ
ステップ (4) で 4 入力加算器出力を符号反転および後処理して FR [n] へ

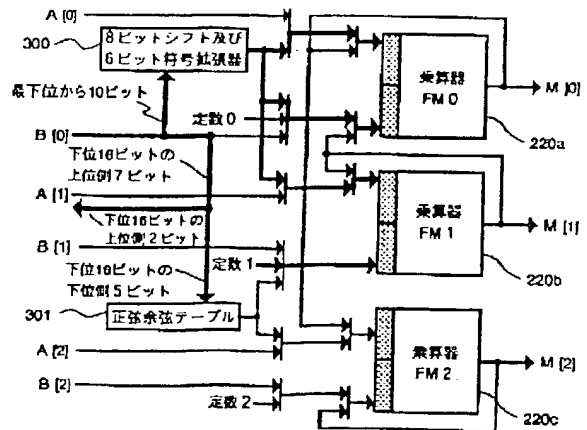
【図 29】

【図 29】



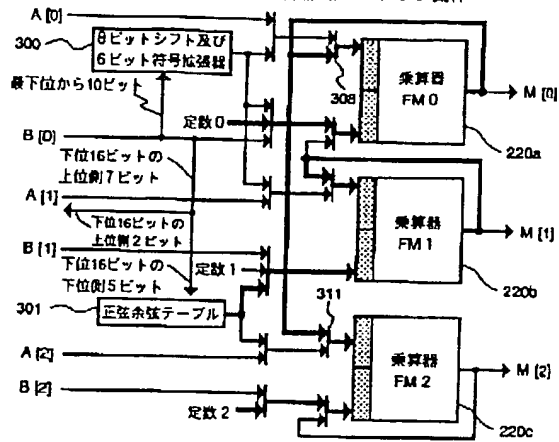
【図 30】

【図 30】 正弦余弦用単精度乗算器第1ステップ動作



【図31】

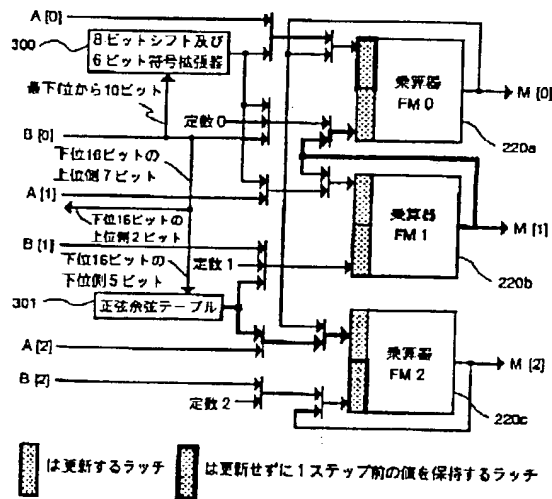
【図31】 正弦余弦用単精度乗算器第2ステップ動作



【図33】

【図33】

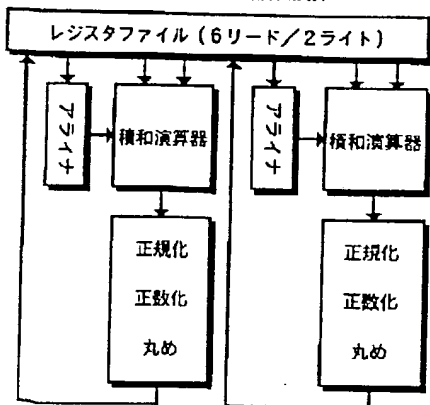
正弦余弦用単精度乗算器第4ステップ動作



【図35】

【図35】

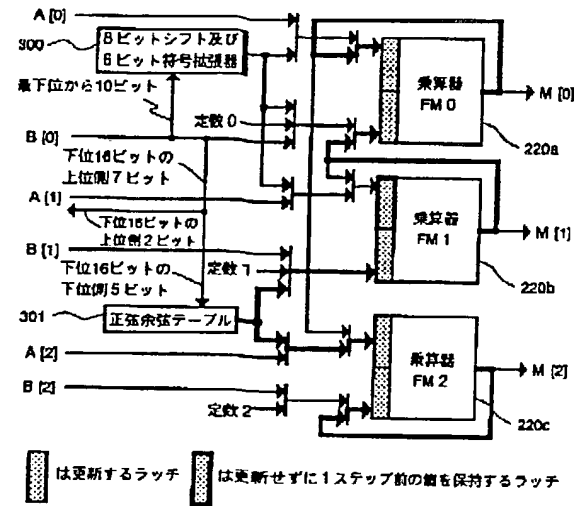
2並列浮動小数積和演算



【図32】

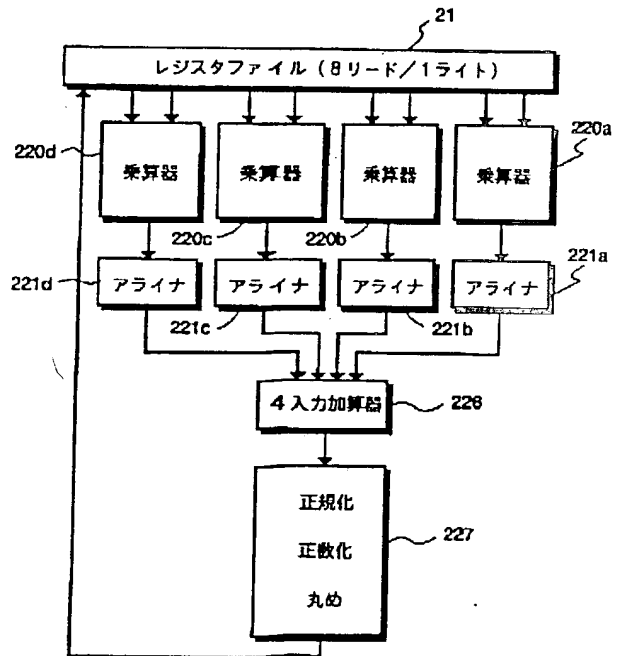
【図32】

正弦余弦用単精度乗算器第3ステップ動作

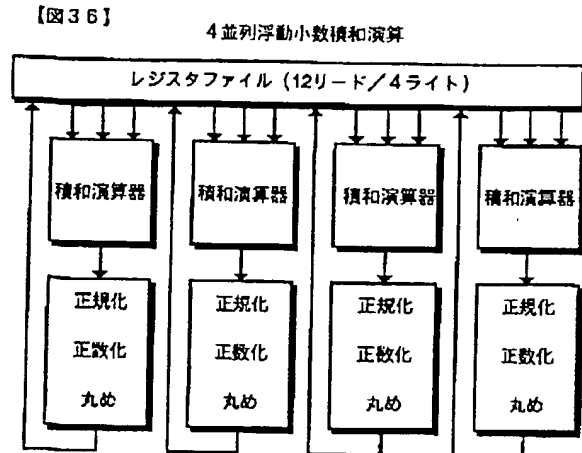


【図34】

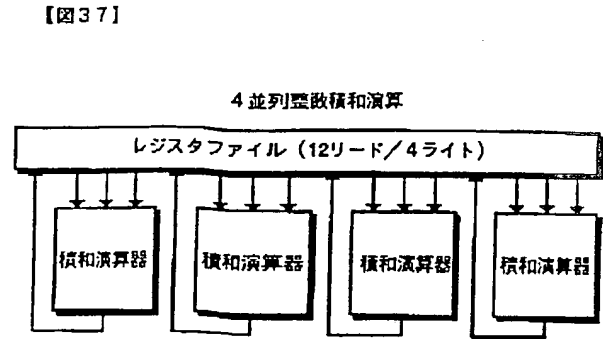
【図34】



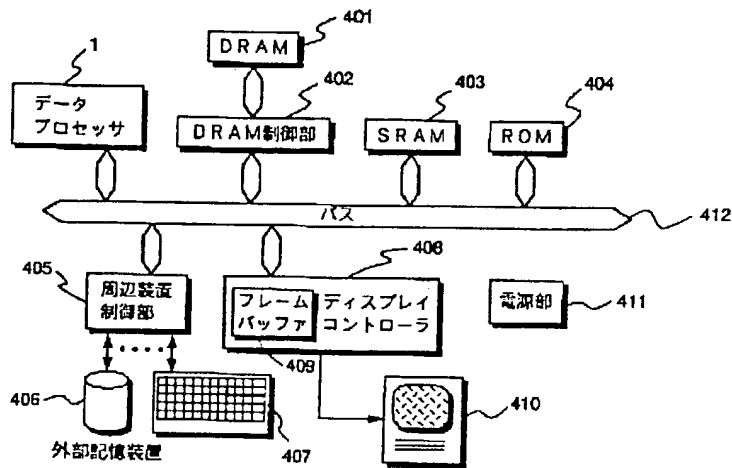
【図36】



【図37】



【図38】



フロントページの続き

(51)Int. Cl. 6

識別記号

F I

// G 0 6 T 1/00

G 0 6 F 15/66

J

(72)発明者 戸塚 米太郎

東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内